PLEASE **AMEND** THE CLAIMS AS FOLLOWS:

1. (Currently Amended) A computer program product for managing execution of an application on a computer according to a lifecycle, the computer program product comprising:

a computer-readable medium storing thereon computer-readable instructions being executable on a computer, the computer-readable instructions including:

instructions for receiving a state change request from the application, the state change request indicating a request from the application that an application manager initiate a change in state of the application from a first state to a second state; and

instructions for initiating the state change of the application by an application manager in response to the state change request received from the application when the second state is an allowable state according to a specified set of rules, thereby enabling the application to initiate its own state change via the state change request without human intervention.

2. (Previously Amended) The computer program product as recited in claim 1, wherein the second state is a paused state or destroyed state.

3. (Previously Amended) A computer-implemented method of managing execution of an application according to a lifecycle, comprising:

receiving a signal indicating that a new service is selected;

initiating execution of the application when the new service is selected such that the application enters an active state;

pausing execution of the application such that the application enters a paused state from the active state;

receiving a resume request from the application that has been paused indicating that the application wishes to resume execution and enter the active state from the paused state; and

starting execution of the application from which the resume request was received such that the application enters the active state from the paused state when the resume request is received from the application, thereby enabling the application to initiate its own state change from the paused state to the active state without human intervention.

4. (Previously Amended) A computer-implemented method of managing execution of a plurality of applications according to a lifecycle, comprising:

initiating execution of each one of the plurality of applications such that the plurality of applications enter an active state;

pausing execution of one of the plurality of applications such that the one of the plurality of applications enters a paused state from the active state;

receiving a resume request from the one of the applications that has been paused, the resume request indicating that the one of the plurality of applications requests to resume execution and enter the active state from the paused state; and

starting execution of the one of the plurality of applications from which the resume request was received such that the one of the plurality of applications enters the active state from the paused state in response to receiving the resume request from the application, thereby enabling the one of the plurality of applications to initiate its own state change from the paused state to the active state without human intervention.

5. (Previously Amended) A computer-implemented method of managing execution of an application according to a lifecycle, comprising:

requesting a first time that the application change its state from a first state to a second state by sending a request to the application, wherein the request is a conditional request that is conditional upon the application's decision to change from the first state to the second state, thereby enabling the application to allow or prevent its own state change from the first state to the second state in response to the conditional request;

determining whether the application has decided to allow its own state change from the first state to the second state in response to the conditional request by ascertaining whether the application has changed its state from the first state to the second state; and

requesting a second time that the application change its state from the first state to the second state when it is determined that the application has not changed its state from the first state to the second state and a predetermined condition is satisfied.

6. (Previously Amended) The method as recited in claim 5, wherein the predetermined condition indicates that a specified period of time has elapsed or that the application is now able to perform the requested state change.

7. (Previously Amended) The method as recited in claim 5, wherein it is determined that the application has not changed its state when a state change exception is raised by the application.

8. (Previously Amended) The method as recited in claim 5, wherein it is determined that the application has not changed its state when the application rejects the requested state change.

9. (Previously Amended) The method as recited in claim 5, wherein it is determined that the application has not changed its state when the application is unable to perform the requested state change.

10. (Previously Amended) A computer-implemented method of managing execution of an application according to a lifecycle, comprising:

requesting that the application change its state from a first state to a second state by sending a request to the application, wherein the request is a conditional request that is conditional upon the application's decision to change from the first state to the second state, thereby enabling the application to allow or prevent its own state change from the first state to the second state in response to the conditional request;

determining whether the application has decided to allow its own state change from the first state to the second state in response to the conditional request by ascertaining whether the application has changed its state from the first state to the second state; and

requesting that the application change its state from the first state to a third state when it is determined that the application has not changed its state from the first state to the second state.

11. (Previously Amended) The method as recited in claim 10, wherein the first state is an active state indicating that the application is currently executing, the second state is a destroyed state indicating that the execution of the application has terminated, and the third state is a paused state indicating that execution of the application has paused such that the application can resume execution.

12. (Previously Amended) The method as recited in claim 10, wherein it is determined that the application has not changed its state when a state change exception is raised by the application.

13. (Previously Amended) The method as recited in claim 10, wherein it is determined that the application has not changed its state when the application rejects the requested state change.

14. (Original) The method as recited in claim 10, wherein it is determined that the application has not changed its state when the application is unable to perform the requested state change.

15. (Previously Amended) An apparatus for managing execution of an application according to a lifecycle, comprising:

a processor; and

a memory, at least one of the processor and the memory being adapted for:

requesting a first time that the application change its state from a first state to a second state by sending a request to the application, wherein the request is a conditional request that is conditional upon the application's decision to change from the first state to the second state, thereby enabling the application to allow or prevent its own state change from the first state to the second state in response to the conditional request;

determining whether the application has decided to allow its own state change from the first state to the second state in response to the conditional request by ascertaining whether the application has changed its state from the first state to the second state; and

requesting a second time that the application change its state from the first state to the second state when it is determined that the application has not changed its state from the first state to the second state and a predetermined condition is satisfied.

16. (Previously Amended) The apparatus as recited in claim 15, wherein the first state is an active, paused, or loaded state and the second state is a destroyed state indicating that the application is terminated.

17. (Previously Amended) The apparatus as recited in claim 15, wherein it is determined that the application has not changed its state when a state change exception is raised by the application.

18. (Previously Amended) The apparatus as recited in claim 17, wherein the second state is an active state indicating that the application is being executed, and the state change exception is raised by the application when the application has entered itself into a paused state or a terminated state.

19. (Previously Amended) The apparatus as recited in claim 15, wherein it is determined that the application has not changed its state when the application rejects the requested state change.

20. (Previously Amended) The apparatus as recited in claim 15, wherein it is determined that the application has not changed its state when the application is unable to perform the requested state change.

21. (Currently Amended) A system for managing execution of an application on a computer according to a lifecycle, the system comprising:

a processor; and

a memory, at least one of the processor and the memory providing:

one or more rules;

an application manager capable of executing one or more applications according to the lifecycle, the lifecycle enabling each of the applications to enter one of a plurality of states in response to one or more associated predetermined commands, the application manager capable of selecting one of the predetermined commands to execute according to the one or more rules; and

a mechanism for enabling the application to at least one of initiate and prevent its own state change from a first one of the plurality of states to a second one of the plurality of states <u>without human intervention</u>.

22. (Original) The system as recited in claim 21, further comprising:

a signaling monitor coupled to the application manager and capable of receiving a data stream, the signal monitor adapted for determining whether an application is present in the data stream and communicating information associated with the application to the application manager.

23. (Original) The system as recited in claim 21, wherein the application manager is configured to store an application context for each of the applications, the application context identifying a current one of the plurality of states.

24. (Original) The system as recited in claim 23, wherein the current one of the plurality of states is identified by the associated application to the application manager.

25. (Original) The system as recited in claim 23, wherein the application context further identifies a class loader capable of loading one or more classes associated with the application.

26. (Original) The system as recited in claim 23, wherein the application context further identifies a display context including display information to be displayed.

27. (Previously Amended) The system as recited in claim 21, further comprising an application environment object enabling the application to communicate with the application manager, thereby enabling the application to at least one of initiate its own state change, inform the application manager of the application-initiated state change, prevent its own state change, and inform the application manager that it is preventing its own state change that has been requested by the application manager.

28. (Previously Amended) The system as recited in claim 23, wherein the application context further identifies an application environment object that enables the application to retrieve properties associated with its runtime environment.

29. (Currently Amended) The system as recited in claim 21, further comprising an application environment object that enables the application to communicate a state change of the application to one of the plurality of states to the application manager, wherein the state change has been initiated by the application without human intervention.

30. (Currently Amended) The system as recited in claim 21, further comprising an application environment object that enables the application to request that the application manager change the current state of the application from a paused state to an active state, thereby enabling the application to initiate its own state change from the paused state to the active state without human intervention.

31. (Original) The system as recited in claim 21, further comprising:

a display manager coupled to the application manager and adapted for managing a display context for each of the applications, the display context being in a first state when the display context is visible and being in a second state when the display context is invisible.

32. (Previously Amended) The system as recited in claim 31, wherein the display context is in the first state when the application is in an active state and in the second state when the application is in a paused state.

33. (Original) The system as recited in claim 31, wherein the state of the display context is determined according to the one or more rules followed by the application manager.

34. (Currently Amended) A digital television receiver for managing execution of an application according to a lifecycle, comprising:

means for determining from a data stream whether an application is present in the data stream;

means for loading an application when it is determined that an application is present in the data stream;

means for executing the loaded application according to the lifecycle, the lifecycle including a plurality of states;

means for enabling the application manager to cause the application to change from one of the plurality of states to another one of the plurality of states; and

means for enabling the application to prevent a change in state of the application requested by the application manager or to communicate to the application manager a state change of the application from one of a first set of the plurality of states to one of a second set of the plurality of states, wherein the state change of the application is initiated by the application without human intervention.

35. (Cancelled)

36. (Currently Amended) The digital television receiver as recited in claim 34, wherein the second set of the plurality of states includes a paused state indicating that the application has been paused and a destroyed state indicating that the application has been terminated, thereby enabling the application to pause or terminate its own execution <u>without human intervention</u>.

37. (Currently Amended) The digital television receiver as recited in claim 34, <u>further comprising:</u>

means for enabling the application to request that the application manager change the state of the application from one of the first set of the plurality of states to one of the second set of the plurality of states, thereby enabling the application to initiate its own state change <u>without human intervention</u>.

38. (Cancelled)

39. (Previously Amended) The digital television receiver as recited in claim 34, wherein the first state is an active state and the second state is a paused state.

40. (Previously Amended) The digital television receiver as recited in claim 34, means for enabling the application to communicate to the application manager that the application cannot change its state as the application manager has requested, thereby enabling the application to prevent its own state change.

41. (Previously Amended) The digital television receiver as recited in claim 40, further comprising:

means for enabling the application to raise a state change exception indicating that the application cannot change its state as the application manager has requested.

42. (Previously Amended) The digital television receiver as recited in claim 34, further comprising:

means for enabling the application to raise a state change exception indicating that the application does not want to change its state as the application manager has requested, thereby enabling the application to prevent its own state change.

43. (Previously Amended) The digital television receiver as recited in claim 36, further comprising:

means for releasing memory associated with the application when the application has been terminated.

44. (Currently Amended) The digital television receiver as recited in claim 34, further comprising:

means for creating a class loader associated with each ~~application~~ one of a plurality of applications such that a plurality of class loaders are generated, each of the plurality of class loaders being ~~class loader is~~ associated with ~~the~~ only one of the plurality of applications~~application~~, the class loader being adapted for loading one or more classes associated with the application;

means for employing the class loader to load the classes associated with one of the plurality of applications ~~the application~~; and

means for instantiating the application using the classes that have been loaded by the class loader.

45. (Previously Amended) The digital television receiver as recited in claim 44, further comprising:

means for unloading the classes associated with the application when the application is terminated.

46. (Currently Amended) The digital television receiver as recited in claim 45, wherein the instructions for unloading the classes comprise:

means for de-referencing the class loader by an application manager.

47. (Previously Amended) The method as recited in claim 5, wherein the second state is terminated.

48. (Previously Added) The method as recited in claim 5, wherein the request includes a parameter, the parameter when in a first state indicating that the state change is conditional and unconditional when in a second state.